flokinet-015 - /32 routes and switch security

Nicholas Morrison nick@nanocat.net

Connecting

SSH to the netlab server:

```
$ ssh flokilab.nanocat.net
```

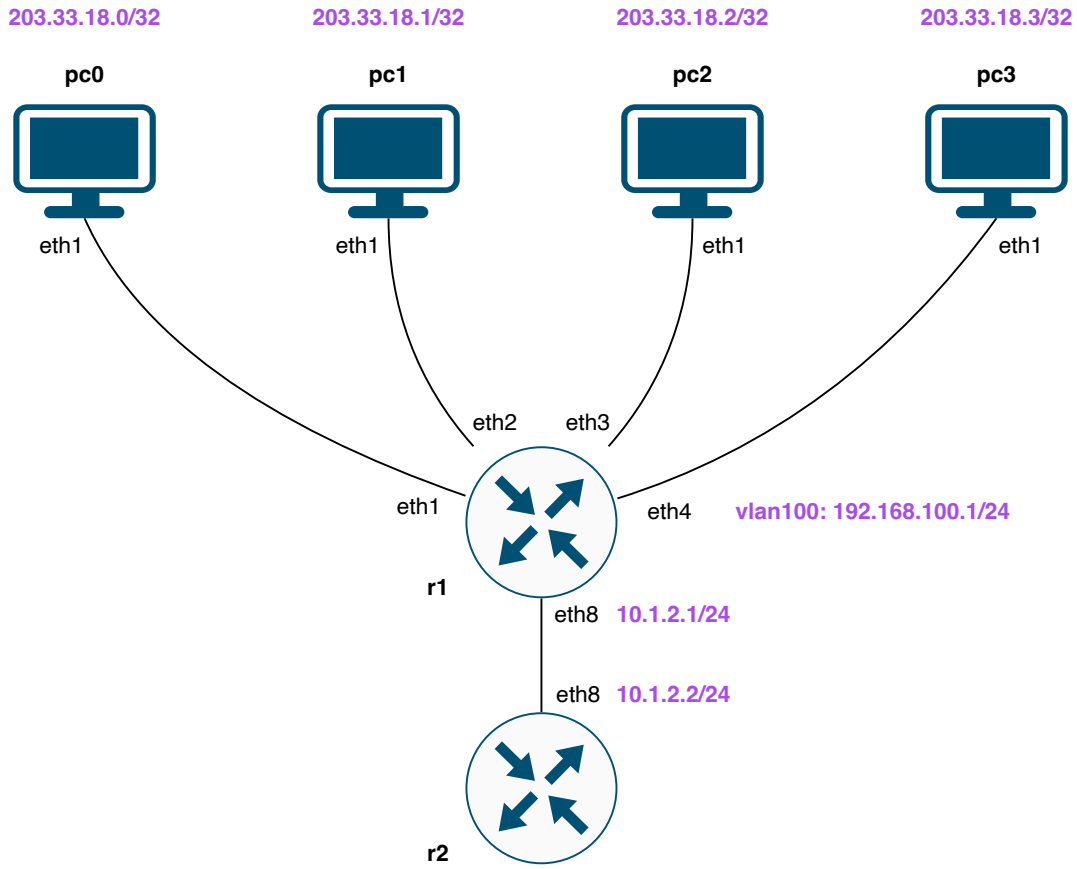
List the running containerlab devices:

```
$ list-devices
```

Connect to a device:

```
$ connect device-name
```

Diagram



Goal

- understand and configure point to point host routes
- understand and configure private VLANs
- understand and configure port-security
- understand and configure static ARP entries
- understand and configure static MAC entries

Design Rules

- different customers should not be able to communicate directly with one another, only via the customer gateway (r1)
- a single IPv4 address should be allocated per customer
- customers should not be able to spoof IP addresses
- customers should not be able to spoof MAC addresses
- you have the network allocation 203.33.18.0/24, from which you can allocate /32s (or larger) to your customers.

Basic configuration

Disable ICMP redirects on r1.

```
! r1
!
no ip icmp redirect
!
```

VLAN configuration

Start by configuring VLAN 100 on r1, into which we will place all customers.

```
! r1
!
vlan 100
!
interface eth1-5
   description Customer devices
   switchport access vlan 100
!
interface vlan 100
   description Customer gateway
   ip address 192.168.100.1/24
!
```

Link between r1-r2

Configure the link from r1 to r2, including OSPF.

```
! r1
!
interface eth8
   description r2
   no switchport
   ip address 10.1.2.1/24
   ip ospf area 0
!
```

```
ip routing
!
router ospf 100
   redistribute static
!

! r2
!
interface eth8
   description r1
   no switchport
   ip address 10.1.2.2/24
   ip ospf area 0
!
ip routing
!
router ospf 100
!
```

Verify that OSPF is working.

```
show ip ospf neighbor
show ip route
```

Configure “the internet”

On r2, configure a loopback interface with the “internet” address `69.69.69.69/32`.

```
! r2
!
interface loopback0
   ip address 69.69.69.69/32
   ip ospf area 0
!
```

Add /32 routes for customers

On r1, add static routes for the customers.

```
!
ip route 203.33.18.0/24 Vlan100
!
```

PC configuration

Configure your four PCs.

- pc0: 203.33.18.0/32
- pc1: 203.33.18.1/32
- pc2: 203.33.18.2/32

- pc3: 203.33.18.3/32
- pc4: 203.33.18.4/32

```
ip address add 203.33.18.0/32 dev eth1
ip route add 192.168.100.1/32 dev eth1
ip route delete default
ip route add default via 192.168.100.1
```

Test with a ping.

```
ping 192.168.100.1
ping 203.33.18.[0-3]
```

Be bad: Try to steal an unallocated IP address

On pc0:

```
ip address add 203.33.18.100/32 dev eth1
```

on pc1:

```
ping 203.33.18.100
```

Protect: remove the /24

On r1:

```
! r1
!
no ip route 203.33.18.0/24
ip route 203.33.18.0/32 Vlan100
ip route 203.33.18.1/32 Vlan100
ip route 203.33.18.2/32 Vlan100
ip route 203.33.18.3/32 Vlan100
ip route 203.33.18.4/32 Vlan100
!
```

Can pc1 still `ping 203.33.18.100`?

Be bad: Set up an unauthorised network between two PCs

On pc2:

```
ip address add 10.100.100.1/24 dev eth1
```

On pc3:

```
ip address add 10.100.100.2/24 dev eth1
```

Ping 10.100.100.1 from pc3. It works, even though we never authorised those IP addresses. PCs in VLAN100 can talk directly with one another.

Protect: configure VLAN isolation

Create a private isolated VLAN:

```
! r1
!
vlan 200
   name customer-isolated
   private-vlan isolated primary vlan 100
!
interface eth1-4
   switchport access vlan 200
!
```

Can pc3 still ping pc2 on the unauthorised IP address?

Can pc3 still ping its default gateway, and 69.69.69.69?

Devices within private isolated VLANs can only communicate with devices in the “parent” primary VLAN.

Be bad: Steal another customer’s MAC address

Somehow, a customer has discovered another customer’s MAC address. Steal it and their IP!

Find the MAC and IP of the victim:

```
# on pc0 (the victim)
ip -c address show dev eth1
# record the inet and the link/ether addresses
```

Steal the MAC and IP of the victim:

```
# on pc1 (the perp)
# first record the original MAC and IP
ip -c address show dev eth1
# clear all IP addresses from eth1
ip address del [all of the IPs] dev eth1
# set the MAC address to the stolen MAC
ip link set dev eth1 address [STOLEN_MAC]
# set the IP address to the stolen IP
ip address add [STOLEN_IP] dev eth1
# ping the internet
ping 69.69.69.69
```

If it doesn’t work right away, try clearing the ARP and MAC tables on r1:

```
show arp
show mac address-table
clear arp [STOLEN_IP]
clear mac address-table
```

Protect: configure port-security

Limit the number of MAC addresses that a switch interface will allow.

Reset pc1's MAC address:

```
# on pc1
# clear all IP addresses from eth1
ip address del [all of the IPs] dev eth1
# set the MAC address to the original MAC
ip link set dev eth1 address [ORIGINAL_MAC]
# set your original IP address
ip address add 203.33.18.1/32 dev eth1
# test - leave this running!
ping 69.69.69.69
```

Clear the ARP and MAC tables on r1:

```
! r1
clear arp [STOLEN_IP]
clear mac address-table
```

Configure port-security on r1:

```
!
interface eth1-4
   switchport port-security mac-address maximum 1
   switchport port-security violation shutdown
!
```

Change the MAC address on pc1:

```
# set the stolen MAC address
ip link set dev eth1 address [STOLEN_MAC]
# try pinging
ping 69.69.69.69
```

Check the status on r1:

```
! r1
show log last 2 minutes
show interface status
show interface status errdisabled
```

That interface is now offline until manually cleared.

Reset the MAC address on pc1, then:

```
! r1
!
interface eth2
   shutdown
   no shutdown
!
```

Set a static ARP and static MAC entry

On r1:

```
!
mac address-table static [PC1_MAC_ADDRESS] vlan 100 interface Ethernet2
!
arp 203.33.18.1 [PC1_MAC_ADDRESS] arpa
!
```